Using a Goal-Agenda and Committed Actions in Real-Time Planning

Damien Pellier, Bruno Bouzy, Marc Métivier
Université Paris Descartes (LIPADE)
45, rue des Saints Pères, Paris, France
{*damien.pellier,bruno.bouzy,marc.metivier*}*@parisdescartes.fr*

***Abstract*—In the context of real-time planning, this paper investigates the contributions of two enhancements for selecting actions. First, the agenda-driven planning enhancement ranks relevant atomic goals and solves them incrementally in a best-first manner. Second, the committed actions enhancement commits a sequence of actions to be executed at the following time steps. To assess these two enhancements, we developed a real-time planning algorithm in which action selection can be driven by a goal-agenda, and committed actions can be done. Experimental results, performed on classical planning problems, show that agenda-planning and committed actions are clear advantages in the real-time context. Used simultaneously, they enable the planner to be several orders of magnitude faster and solution plans to be shorter.**

***Keywords*-Artificial Intelligence, Automated Planning, Real-time Search**

I. Introduction

Real-time planning (*RTP*) is a longstanding goal of the planning community: many applications such as robot task planning, process control, video games require real-time planning. The aim of our research is to tackle the problem of *RTP* with a hard real-time criterion in its deterministic form [1] to solve task planning problem expressed in STRIPS [2] (real-time planning was also generalized to non-deterministic problems by [3] and to probabilistic problems by [4]).

In order to address RTP, various techniques have been devised to gain real-time performance. The most important technique is based on Real-Time Search (*RTS*), e.g., [1]. This technique has been strongly linked to the development of video games and path-finding. There are several real-time search algorithms, e.g., mini-min [1], *B-LRTA** [5], γ-Trap and *LRTS* [6], *LRTA**$_{LS}$ [7], or *LSS-LRTA** [8]. The better the action selection, the better the overall solution plan. Since the action selection time is limited, these algorithms explore a small part of the state space around the current state, and the plans executed are not necessarily optimal. There is much room for improving the action selection process.

To this extent, the goal agenda technique, originally developed for classical planning but not for real-time planning, seems worth considering. This technique, devised first by [9] and extended by [10], [11], [12], [13], has been successfully used in plan generation to speed up the search. The general idea of this technique can be summarized as follow: Given a set of atomic goals, the aim is to compute a *goal agenda* as a total ordering between increasing subsets of goal atoms. Then, planning is viewed as an incremental search for increasing goal subsets from the goal agenda. This way, the efficiency of planning can be dramatically improved in many cases while preserving the completeness of the planning algorithm. As far as we know, the goal agenda technique has not been used in on-line planning yet.

In this paper, we investigate the contribution of two enhancements for selecting actions in real-time context. Enhancement *I*, also called incremental planning or agenda-driven planning, ranks relevant atomic goals and solves them incrementally in a best-first manner, and enhancement *C*, also called committed actions enhancement, commits a sequence of actions to be executed at the next time steps. To assess these two enhancements, we developed a real-time planning algorithm, called Learning Real-Time Planning (*LRTP*) based on the *LRTS* template [6]. Committed actions (*C*) can be included or not. Incremental planning (*I*) can be used or not.

The paper is organized as follows: section 2 gives an overview of the *RTP* planning literature, section 3 shows the goal-agenda driven planning literature and section 4 introduces the notations and the pseudo-code of *LRTP* with enhancements *I* and *C*. Section 5 yields the experimental results of our approach, and shows that agenda-driven planning methods and committed actions can be successfully associated by real-time planning algorithms. Before concluding, section 6 discusses the results.

II. Real-Time Planning

RTP is considered in the context of agent-centered search. While classical off-line search methods first find a solution plan, and then make the agent execute the plan, agent-centered search interleaves planning and execution. The agent repeatedly executes a task, called an episode or a trial. At each step of an episode, the agent is situated in a current state, and performs a search within the space around the current state in order to select its best action. The states encountered during the action selection stage are called the explored states. Conversely, the states encountered by the agent are called the visited states.

The feature of *RTP* is that each local search has a fixed *time budget* that cannot be exceeded. When the action is

selected, the agent executes the action, and reaches a new state. When the agent reaches the goal state, another trial or episode can be launched. Repeating episodes has two interests for real-time planning. First, it allows to obtain meaningful empirical results when action selection includes randomness to break ties. It is the case in our approach. Second, when the agent is able to learn, i.e., the agent updates the heuristic value of the encountered states when some conditions take place, the efficiency of the agent increases over repetitions until the optimal solution is found. Without learning, the efficiency of the agent is based only on the ability of the search to select an action.

The fundamental paper of *RTS* is Real-Time Heuristic Search [1]. Real-Time *A** (*RTA**) computes a tree as *A** does, but in constant time. The learning version of *RTA** is called *LRTA**. Other works in *RTS* worth considering are FALCONS [14], weighted *A** [15], γ-Trap and *LRTS* [6], and recently *LSS-LRTA** [8].

FALCONS [14] is a learning algorithm that converges quickly under some assumptions. Its main feature is to compute two heuristic functions, one for each way from start to goal, and from goal to start.

In weighted *A** [15], the heuristic function has a weight $1+\epsilon$. The greater ϵ the most important the heuristic function. The risk is that the heuristic function becomes non admissible. Nevertheless, even with non admissible functions, the heuristic search finds good plans, although not optimal.

LRTS [6] is a unifying framework for learning in real-time search, including γ-Trap and the principles of *LRTA** and *SLA**. γ-Trap includes some lookahead to smooth the bad effects of the heuristic function. *SLA** [16] includes backtracking. Finally, Bulitko [17] describes dynamic control in real-time heuristic search.

III. Agenda-Driven Planning

One way in which search can be informed concerns the order in which planning atomic goals or sub-goals of a planning problem should be addressed. This technique, called agenda-driven planning, can improve the search by helping the planner to focus on a progressive path toward a solution. An ideal order is one in which each sub-goal does not invalidate any previously achieved sub-goals. If each sub-goal does not invalidate any previous sub-goals during planning, then agenda-driven planning algorithm effectively decomposes the initial goals set into a sequence of sub-problems, and then can resolve each sub-problem incrementally by adding a small number of sub-goals. Conversely, if many sub-goals are invalidated after being made true, the agenda-driven planning algorithm becomes inefficient, because the previous efforts to achieve sub-goals will be wasted by their invalidation. Consequently, the efficiency of an agenda-driven planning algorithm depends on the quality of the goal-agenda which drives the search.

The problem of building a goal-agenda can be summarized as the following problem: given a set of conjunctive goals, can we define and detect an ordering relation over subsets from the original goal set? One of the first and most significant work about this question was proposed by [9]. They introduced the notion of *reasonable orders* which states that a pair of goals A and B can be ordered so that B is achieved before A if it is not possible to reach a state in which A and B are both true, from a state in which only A is true, without having to temporarily destroy A. In such a situation it is reasonable to achieve B before A to avoid unnecessary effort.

Three main extensions of this preliminary work were proposed. The first one, called landmarks planning, was introduced by [11]. It does not only order the (top-level) goals, but also the sub-goals that will necessarily arise during planning, i.e., it also takes into account what they called the "landmarks". The key feature of a landmark is that it must be true at some point on any solution path to the given planning tasks. This approach was successfully implemented in the planner LAMA [13]. The second extension was presented by [12]. This approach consists in three steps: (1) building a planning graph G [18] from the initial state until a fixed point is reached without computing the mutual exclusions, (2) based on G, extracting a relaxed plan by ignoring the delete effects as in FF's relaxed-plan heuristic [10] and (3) determining a proper order between each pair A and B of sub-goals by examining all actions in G that make B true. An important property of this approach is that all partial orders detected by *reasonable ordering* are also detected by this approach. Thus, this approach is strictly stronger that the previous one.

Finally, [19] has recently introduced an incremental search algorithm and a search guidance heuristic for planning with temporally extended goals and uncontrollable events.

IV. LRTP with Agenda and Committed Actions

This section gives the notations and the implementation of *LRTP*.

A. Notations

We consider sequential planning in the propositional STRIPS framework [2]. All sets are assumed to be finite. A *state* s is a set of logical propositions. An *action* a is a triplet $a = (pre(a), add(a), del(a))$ where $pre(a)$ are the action's *preconditions*, $add(a)$ is its positive *effects* and $del(a)$ its negative ones, each a set of propositions. The result of applying a single action a, noted as a sequence of one action, to state s is defined by the transition function γ as follows:

$$\gamma(s, \langle a \rangle) = \begin{cases} (s - del(a)) \cup add(a) & \text{if } pre(a) \subseteq s \\ \text{undefined} & \text{otherwise} \end{cases}$$

The result of applying a plan π as a sequence of more than one action $\pi = \langle a_1, \ldots, a_n \rangle$ to a state s is recursively defined as $\gamma(s, \langle a_1, \ldots, a_n \rangle) = \gamma(\gamma(s, \langle a_1, \ldots, a_{n-1} \rangle), \langle a_n \rangle)$. Applying an empty plan does nothing, i.e., $\gamma(s, \langle\rangle) = s$. A *planning problem* (A, s_0, g) is a triplet where A is the set of actions, s_0 the initial state and g the set of goals. A plan $\pi = \langle a_1, \ldots, a_n \rangle$ with $a_1, \ldots, a_n \in A$ is solution for a planning problem (A, s_0, g) if $g \subseteq \gamma(s_0, \pi)$. The time allocated to action selection is upper-bounded by a constant t_d called the *decision time*. t_g is the global time allocated to perform the episodes.

Since the focus of the current work is action selection, learning is not presented and not shown in the algorithms introduced in the next section. Indeed, learning is not one of our contributions because we use a classical learning mechanism already implemented in [6].

B. Algorithm

LRTP is shown in algorithm 1. It takes a planning problem (A, s_0, g), the decision time t_d and the global time t_g as input parameters. *LRTP* enters the episode loop (line 1) where it iteratively executes episodes while time is running. s_0 is the initial state from which planning and execution start at the beginning of an episode. s is the current state from which planning is computed, and s_r is the current state in the world in which actions are executed (line 2). Then *LRTP* enters the decision loop (line 3). At each iteration, *LRTP* searches the best sequence of actions $\pi' = \langle a_1, \ldots, a_n \rangle$ leading to the goal g. The search is done by *IASA** when enhancement *I* is on (line 4) and by *ASA** otherwise (line 5).

When enhancement *C* is on (line 7), *LRTP* appends the sequence of actions π' to the current plan π and updates s with $\gamma(s, \pi')$. To this extent, *LRTP* jumps from its current state up to the state following the last action of the sequences of actions which becomes its new current state. When enhancement *C* is off (line 8), *LRTP* appends the first action a_1 of the sequence of actions π' to the current plan π of the episode and updates s with $\gamma(s, \langle a_1 \rangle)$. Then, enhancement *C* being on or off, *LRTP* gets the first action of plan π, executes this action and removes it from π. Finally, s_r is updated (line 10).

When enhancement *C* is on, *LRTP* cannot change the actions put into its buffer. As a result, s_r and s can be different. *LRTP* takes advantage of the fact that several actions are committed in a single decision time in order to accumulate a credit of time for the next time steps. This technique enables *LRTP* to search deeper into the local search space and to increase the quality of the sequence of actions computed.

C. Action Selection

The aim of action selection is to return a sequence of actions depending on the given decision time. It is based on *A** [20]. *ASA** is the action selection without incremental planning. *ASA** is shown in Algorithme 2. When t_d is elapsed, the set S_{open} of open states is considered (line 1). The procedure selects the set S_f of states with the lowest f-value (line 2). Then, it selects the set S_g of states with the lowest cost (line 3). Then, the procedure applies a random choice to select the best state s (line 4). The idea is to give priority to the state closer to the initial state in order to increase the robustness of the returned sequence of actions. This technique is not new in RTS [6].

*IASA** corresponds to action selection with incremental planning. The *IASA** strategy is given in Alg. 3. It uses the goad-agenda to drive its search. First, the local planning search procedure computes the goal-agenda, based on the relaxed plan ordering proposed by [12], by determining all ordering relations that hold between atomic goals (line 2). The goal g is now a sequence of atomic propositions $\langle g_1, \ldots, g_n \rangle$ such as $g = \bigcup_{i=1}^{n} g_i$. Then, for each atomic goal g_j and as long as the time allocated to the local search is not exceeded, the local planning procedure searches for a plan for an initial state s_i (initially set to s_0) with the *ASA** procedure, and the incremental goal $g_i = \bigcup_{j=1}^{i} g_j$ (line 4-5). The solution plan for g_i is added to the sequence of actions under construction (line 6) and the state s_i is updated with $\gamma(s_i, \pi')$ (line 7). Finally, if all atomic goals are reached or if the decision time is exceeded, the search stops and the procedure returns the sequence of actions computed.

Finally, if the *ASA** search fails then the local planning search also fails and the episode ends.

Algorithm 1: LRTP(A, s_0, g, t_d, t_g)

```
1  while elapsed_time < t_g do
2      π ← ⟨⟩, s ← s_0, s_r ← s_0
3      while elapsed_time < t_g and g ⊈ s_r do
4          if I is on then π' ← IASA*(A, s, g, t_d)
5          else π' ← ASA*(A, s, g, t_d)
6          Let π' = ⟨a_1, ..., a_k⟩
7          if C is on then π ← π · π', s ← γ(s, π')
8          else π ← π · a_1, s ← γ(s, ⟨a_1⟩)
9          Execute and remove the first action a of π
10         s_r ← γ(s_r, ⟨a⟩)
```

Algorithm 2: ASA*(A, s_0, g, t_d)

```
1  S_open ← A*(A, s_i, g_i, t_d)
2  Let S_f be the set of states with the lowest f of S_open
3  Let S_g be the set of states with the lowest g of S_f
4  Choose randomly a state s from S_g
5  return the plan from s_0 to s
```

Algorithm 3: IASA*(A, s_0, g, t_d)

$\pi \leftarrow \langle\rangle$, $i \leftarrow 1$, $s_i \leftarrow s_0$
$\langle g_1, \ldots, g_n \rangle \leftarrow$ RelaxedPlanOrdering(A, s_0, g)
while $elapsed\_time < t_d$ *and* $i \leq n$ **do**
  $g_i \leftarrow \bigcup_{j=1}^{i} g_j$
  $\pi' \leftarrow$ ASA*$(A, s_i, g_i, t_d - elapsed\_time)$
  $\pi \leftarrow \pi \cdot \pi'$
  $s_i \leftarrow \gamma(s_i, \pi')$
  $i \leftarrow i + 1$
return π

V. EXPERIMENTS

The objective of these experiments is to evaluate the performances of *LRTP*, with or without enhancement *I*, and with or without enhancement *C*, in classical planning problems taken from the International Planning Competition (IPC). We use the non-admissible heuristic function of FF [10] to drive the search. The use of non-admissible heuristics for real-time search is not new [15]. We have four algorithms to assess: *LRTP* without enhancement, *LRTP* with incremental planning (*LRTP+I*), *LRTP* with committed actions (*LRTP+C*), and *LRTP* with both enhancements (*LRTP+IC*).

A. Description, settings and performance measures

Any experiment consists in running each algorithm in a specific problem with a defined decision time and for a defined number of episodes. The performances are computed according to the performance indice used in IPC-8[1] which measures the quality of the solutions provided by the algorithms. Consider a problem with an optimal solution plan having Q^* actions. At the end of each episode, an algorithm receives the score Q^*/Q if it has found a plan having Q actions, or zero if it has not found a plan. Then the performance of this algorithm in the problem is computed as the average all scores received[2]. In cases where the optimal plan length of a problem is not known, Q^* will be considered the best plan length found by the algorithms in that problem.

In a first stage, the four algorithms are assessed in a specific problem taken from IPC-5, the 15th problem of the Rovers domain, chosen in order to highlight how the different algorithms behave as a function of the decision time. All experiments involve 10 episodes. All episodes end if the algorithm reaches the goal or have executed 400 actions.

In a second stage, the algorithms are evaluated on all the problems of Rovers domain with different decision times. Each experiment contains 100 episodes. Episodes terminate if the algorithm reaches the goal or if 500 actions have been executed. The results provided are the mean performances on each problem of the domain.

Finally, in a third stage, we present the results obtained by the four algorithms on all the problems of 15 domains. A specific decision time is used for each domain. As previously each experiment contains 100 episodes and any episode terminates if the algorithm reaches the goal or if 500 actions have been executed. The results provided are the mean performances on each domain.

All the tests were conducted on an Intel Core 2 Quad 6600 (2.4Ghz) with 4 Gbytes of RAM.

[1] http://ipc.informatik.uni-freiburg.de/EvaluationSatisficing

[2] The original definition of IPC-8 performance indice uses a sum instead of the mean. We decide to use the mean in order to facilitate performance comparisons from an experiment to another, especially when we consider several domains

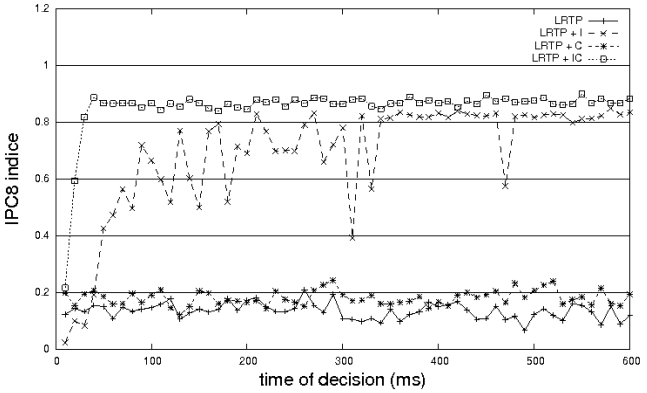

Figure 1. IPC8 indice depending on the decision time (IPC5 Rovers pb. 15).

B. Rovers 15

This subsection presents the results obtained by the four algorithms in the fifteenth problem of Rovers domain, Rovers 15.

Figure 1 shows the IPC8 indice of each algorithm according to the decision time. In these results Q^* is set to 39 which is the length of the optimal plan in Rovers 15.

Best performances were clearly obtained when using both enhancements. As a matter of fact, the performances of *LRTP+IC* was close to 0.87 with all decision time up to 40 ms. With such decision time, *LRTP+IC* reached the goal in every episodes and developed plans that was about 45 actions length.

Parallely, *LRTP+I* needed at least 340 ms of decision time to obtain performances almost always higher than 0.8. As *LRTP+IC*, plans was found in every episodes. The plans developed contained about 48 actions.

Finally, *LRTP* and *LRTP+C* obtain very poor performances whatever the decision time we used. In fact, these algorithms most often reached the goal but they developed plans having more than 200 actions.

C. Rovers domain

This subsection presents the results obtained by the four algorithms on all problems of Rovers domain. It shows

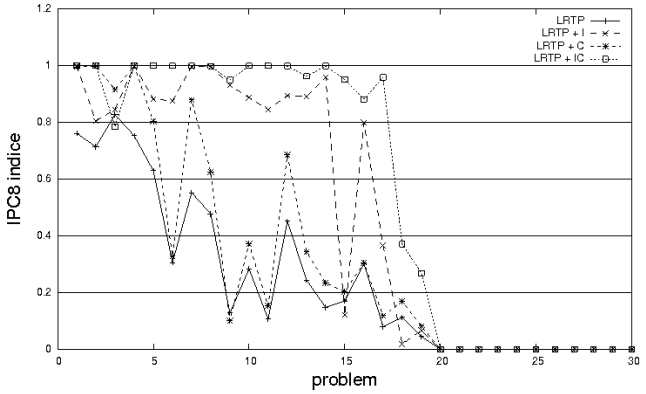

Figure 2. IPC8 indice in Rovers domain depending on the problem when the decision time is set at 100ms.

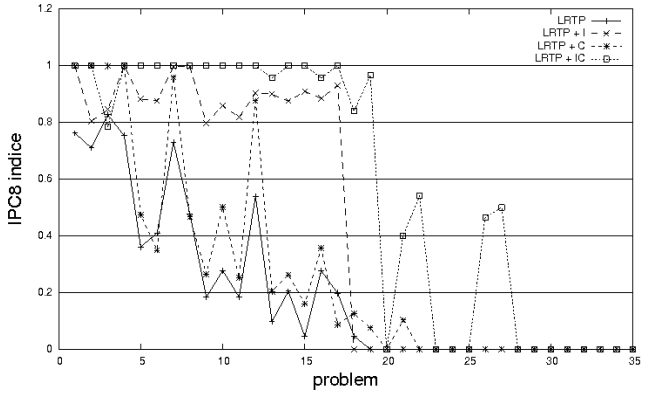

Figure 3. IPC8 indice in Rovers domain depending on the problem when the decision time is set at 1000ms.

the IPC8 indice obtained in each problem. Two cases are considered: a first case where the decision time is set at 100ms, and a second case where the decision time is set at 1000ms.

Figure 2 shows the IPC8 indice of each algorithm over the 40 problems of Rovers domain when the decision time is set at 100ms. First we can see that in the first 19 problems, all algorithms have found a solution plan in at least one episode for each problem. From the 20th problem, no solution plan has been found by the algorithms.

Without enhancement, we can see that the planner developed rather poor performances. Their performances were up to 0.6 in problems 1 to 5, but decreased rapidly in the following problems, being less than 0.5 from problem 10.

With enhancement *C*, the results show similar behaviors to the ones obtained without enhancement but with globally better performances. High performances, up to 0.8, have been obtained in the first 5 problems and in the 7th one. In Rovers 8 and 12, the performances were up to 0.6, while in other problems, they were all less than 0.4.

With enhancement *I*, the performances are up to 0.8 until the 14th problem. They stay close to 0.8 in problem 16, and less than 0.4 in all problems left.

Finally, the best performances have been obtained with the two enhancements. They stay up to 0.8, mostly up to 0.95, until problem 17 and then decrease rapidly. One exception is observed in problem 3 where *LRTP+IC* reached 0.79 which is the worst performance obtained by the algorithms in that problem. In that specific case, *LRTP+C* have reached the best performance with a value close to 0.9, followed by *LRTP+I*, *LRTP+I* and finally *LRTP+IC*, all last three having performances close to 0.8. However, it is worth mentioning that problem 3 is a very simple problem for the four algorithms. In fact, the four algorithms found a solution plan in every episode. The differences observed in the results come from the fact that the mean plan lengthes obtained by the algorithms are respectively 12, 13, 13.29 and 14.

Figure 3 shows the IPC8 indice of each algorithm over the 40 problems of Rovers domain when the decision time is set at 1000ms. Until the 14th problem, we observed results very close to the one observed when the decision time is set to 100ms. *LRTP+IC* was still the best algorithm followed by *LRTP+I*, *LRTP+C* and *LRTP* and, as previously, one exception was observed in problem 3 where *LRTP+C* is the best algorithm followed by *LRTP+I*, *LRTP+I* and finally *LRTP+IC*.

Beyond problem 14, we can observe that having 1000ms of time to take a decision did not enable *LRTP* and *LRTP+C* to develop better plans. On the other side, we can observe that it enabled *LRTP+I* and *LRTP+IC* to improve their performances. As a matter of fact, *LRTP+I* reached performances close to 0.9 until problem 17. With *LRTP+IC*, the performances stayed up to 0.95 until problem 17. In problems 18 and 19, *LRTP+IC* respectively reached 0.84 and 0.97. Using 1000ms of decision time enabled *LRTP+IC* to find a solution plan in problems 21, 22, 26 and 27.

D. Results in several domains

This subsection presents the results obtained by the four algorithms on all problems of 15 domains issued from several competitions : Airport (IPC-4), Blocksworld (IPC-2), Depot (IPC-3), Driverlog (IPC-3), Elevator (IPC-2), Freecell (IPC-3), Logistics (IPC-2), Openstacks (IPC-5), Pathways (IPC-5), Pipesworld (IPC-5), PSR (IPC-4), Rovers (IPC-5), Satellite (IPC-4), TPP (IPC-5) and Zenotravel (IPC-3). In each domain, we set the decision time to a specific value and ran the algorithms in all the problems of the domain. Then, for each algorithm in each domain, the performance measures provided are averages of the performances obtained in each problems of the domain. Each experiment contains 100 episodes. Episodes terminate if the algorithm reaches the goal or if 500 actions have been executed.

The decision time used depends on the domain considered. It has been set high enough to allow the algorithm to

Domain	LRTP	LRTP+I	LRTP+C	LRTP+IC
Airport	0.22	+129 %	+30 %	+131 %
Blocksworld	0.25	+21 %	+153 %	+135 %
Depot	0.22	-10 %	+60 %	+84 %
Driverlog	0.13	-16 %	+294 %	+306 %
Elevator	0.71	+21 %	+28 %	+23 %
Freecell	0.24	+81 %	+100 %	+101 %
Logistics	0.35	+61 %	+42 %	+137 %
Openstacks	0.68	-32 %	+8 %	+10 %
Pathways	0.11	+228 %	+36 %	+436 %
Pipesworld	0.28	-72 %	+69 %	+80 %
PSR	0.52	+37 %	+27 %	+37 %
Rovers	0.18	+116 %	+34 %	+188 %
Satellite	0.20	-9 %	-0.5 %	+96 %
TPP	0.18	+18 %	+6 %	+65 %
Zenotravel	0.41	+21 %	+55 %	+92 %

Table I
COMPARISON OF IPC8 INDICE DEPENDING ON THE DOMAINS.

Domain	#prob	LRTP	LRTP+I	LRTP+C	LRTP+IC
Airport	50	15	26	15	31
Blocksworld	35	27	11	31	28
Depot	22	10	7	13	11
Driverlog	20	15	3	15	15
Elevator	150	150	150	150	150
Freecell	60	46	32	39	39
Logistics	28	18	26	22	27
Openstacks	30	26	23	25	25
Pathways	30	8	20	5	25
Pipesworld	50	30	7	35	35
PSR	50	36	39	39	40
Rovers	40	18	17	20	23
Satellite	36	16	10	11	18
TPP	30	8	9	8	14
Zenotravel	20	19	15	20	20

Table II
NUMBER OF SOLVED PROBLEMS ACCORDING TO THE DOMAINS.

tackle a majority of the problems of each of the domains. Thus, it has been set at 100ms in domain Driverlog, 200ms in Pathways, 500ms in domains Blocksworld, Depots, Elevator, Freecell, Logistics, Pipesworld-no-Tankage, Satellite and Zenotravel, and finally at 1000ms in Airport, Opentrack, PSR, Rovers and TPP.

Table I shows the average IPC8 indice of LRTP for each domain studied and the percentage of increase or decrease observed when the algorithm uses each enhancement. We can see that, in the majority of the domains, the best performances are obtained when using the two enhancements. Some exceptions are observed in Blocksworld and Elevator, where using enhancement *C* alone gives slightly better performances than when using the two enhancements. It can also be observed that using enhancement *C* always improves the performances, with or without using enhancement *I*, excepted in Satellite where, in this specific case, using one of the enhancements decreases the performances while using the two together results in the best performance. Finally, using enhancement *I* alone decreases the performance of LRTP in Depot, Driverlog, Openstacks, Pipesworld and Satellite.

In addition to these results, table II shows the number of problems solved by each algorithm in each domain. A problem is considered as solved by an algorithm if a solution plan is found in at least one episode during the experiment. These results illustrate that the performance increases are most often associated with a higher number of solved problems. However, some results highlight the ability of the enhancements to increase the quality of the solution plans developed. As a matter of fact, we can observe in Driverlog that *LRTP*, *LRTP+C* and *LRTP+IC* all solved 15 problems among the 20 problems of the domain, but showed high differences between the performance indices: *LRTP+C* and *LRTP+IC* increased the performance of *LRTP* of more than 200%. Similarly, we can observe in Elevator that all algorithms have found a solution plan in the 150 problems of the domain. However, applying at least one of the enhancements enabled to improve the performance of LRTP by 20 to 30%. In Pipesworld and Zenotravel, *LRTP+IC* solved as many problems as *LRTP+C* while having better global performances. In these two cases, this result is due to fact that *LRTP+IC* found better plans than *LRTP+C*. Finally, we can observe that *LRTP+I* solved less problems than *LRTP* in Zenotravel while having 21% of performance improvement. As previously, this result is obtained in consequence to the fact that *LRTP+I* developed better plans than *LRTP* in the majority of the problems it solved.

VI. DISCUSSION

First, we shall comment upon the global observations given by tables I and II. *LRTP+I* is not clearly an improvement over *LRTP* in terms of plan quality and problems solved. This can be explained by the fact that *LRTP* adopts a global view of the problem to solve. *LRTP+I* follows the informations of its goal agenda, and has a very local view of what to do. On domains in which *LRTP+I* is inferior to *LRTP*, the goal agenda does not contain a satisfactory ordering of goals and the performances decrease.

LRTP+C is clearly an improvement over *LRTP* in terms of plan quality and problems solved. With small decision times (sufficiently small for *LRTP* to be unable to find a solution), acting rapidly is a good strategy. *LRTP+C* commits a sequence of actions in its buffer. This sequence can be good or not, provided it is given quickly. Generally, the sequence is far from optimal. The effect of committing the sequence is to free computing time for the future time steps, and therefore to find better solutions later.

LRTP+IC is a clear improvement over *LRTP+C* (and of course over *LRTP+I* and *LRTP*). The decision time being short, you have to commit actions quickly, and basic actions resulting from atomic goals are better than complex actions resulting from global and complex planning considerations. This shows that when using enhancement *C*, it is better to

commit sequences of actions that are simple (provided by the goal agenda). Table I shows that *LRTP+IC* provides solution plans that are as good as the plans provided by *LRTP+C* and table II that *LRTP+IC* solves more problems than *LRTP+C*. Consequently, *LRTP+IC* provides quickly better plans, and solves a larger range of problems. This shows that the two enhancements *I* and *C* must be used simultaneously.

Second, *LRTP* committed actions are different from *LRTS* jumps [6]. When a sequence of actions is found, *LRTS* assumes that the execution of the whole sequence of actions is possible during the same time step. However, this assumption is not always permitted, and *LRTP* commits itself to execute the actions contained in the sequence at the next time steps, one by one. No time gain is observed directly. However, committing several actions to be executed later frees computing time for the action selection process in the following time steps, which results in finding better solution plans, and indirectly saves time.

VII. Conclusion

In this paper, we presented a real-time planning framework called *LRTP* to assess the contribution of two enhancements: incremental planning (*I*) and committed actions (*C*). Enhancement *I* speeds up the action selection process but decreases the percentage of success. Enhancement *C* keeps the percentage of success unchanged and does not decrease the length of the solution plans. The strong point of our work is that, with the two enhancements used simultaneously, *LRTP* obtains results surpassing the results of other versions of *LRTP*. The percentage of success is better, the solution plans are shorter, and the time to execute them is shorter too. Furthermore, the range of the decision time in which *LRTP* acts with the two enhancements decreases by several orders of magnitude compared to the range of the decision time in which *LRTP* acts without enhancement (few hundreds of milliseconds down to few milliseconds). As far as we know, this is the first time that someone integrates with success a goal agenda into an action selection process within real-time planning. The committed actions idea is less innovative but makes the results of the incremental planning idea significant.

Various interesting research directions are now open. First, it would be interesting to highlight learning in our framework: not only to learn the heuristic function used to drive the search, but also to learn goal-agenda heuristics. Second, it would be interesting to propose and compare other action selection strategies such as landmarks planning [13], or stratified planning [21]. Third, broadening the test set would also be useful.

VIII. Acknowledgments

This work has been supported by French National Research Agency (ANR) through COSINUS program (project EXPLO-RA number ANR-08-COSI-004).

References

[1] R. Korf, "Real-Time Heuristic Search," *Artificial Intelligence*, vol. 42, no. 2-3, pp. 189–211, 1990.

[2] R. Finke and N. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," *Artificial Intelligence*, vol. 3-4, no. 2, pp. 189–208, 1971.

[3] S. Koenig and R. Simmons, "Real-time search in non-deterministic domains," in *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995, pp. 1660–1669.

[4] A. Barto, S. Bradtke, and S. Singh, "Learning to Act Using Real-Time Dynamic Programming," *Artificial Intelligence*, vol. 72, no. 1-2, pp. 81–138, 1995.

[5] B. Bonet, G. Loerincs, and H. Geffner, "A Robust and Fast Action Selection Mechanism for Planning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 1997, pp. 714–719.

[6] V. Bulitko and G. Lee, "Learning in Real-Time Search: A Unifying Framework," *Journal of Artificial Intelligence Research*, vol. 25, pp. 119–157, 2006.

[7] C. Hernández, S. Concepción, and P. Meseguer, "Lookahead, Propagation and Moves in Real-Time Heuristic Search," in *Proceedings of the International Symposium on Combinatorial Search*, 2009.

[8] S. Koenig and X. Sun, "Comparing Real-Time and Incremental Heuristic Search for Real-Time Situated Agents," *Journal of Autonomous Agents and Multi-Agent Systems*, vol. 18, no. 3, pp. 313–341, 2009.

[9] J. Koehler and J. Hoffmann, "On Reasonable and Forced Goal Orderings and their Use in an Agenda-Driven Planning Algorithm," *Journal of Artificial Intelligence Research*, vol. 12, pp. 339–386, 2000.

[10] J. Hoffmann and B. Nebel, "The FF Planning System: Fast Plan Generation Through Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 14, no. 1, pp. 253–302, 2001.

[11] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered Landmarks in Planning," *Journal of Artificial Intelligence Research*, vol. 22, pp. 215–278, 2004.

[12] C.-W. Hsu, B. Wah, and Y. Chen, "Subgoal Ordering and Granularity Control for Incremental Planning," in *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2005, pp. 507–514.

[13] S. Richter and M. Westphal, "The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks," *Journal of Artificial Intelligence Research*, vol. 1, no. 39, pp. 127–177, 2010.

[14] D. Furcy and S. Koenig, "Speeding up the Convergence of Real-Time Search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2000, pp. 891–897.

[15] M. Shimbo and T. Ishida, "Controlling the Learning Process of Real-Time Heuristic Search," *Artificial Intelligence*, vol. 146, no. 1, pp. 1–41, 2003.

[16] L. Shue and R. Zamani, "An Admissible Heuristic Search Algorithm," in *Proceedings of the International Syposium on Methodologies for Intelligent Systems*, ser. LNAI, no. 689. Springer, 1993, pp. 69–75,.

[17] V. Bulitko, M. Lustrek, J. Schaeffer, Y. Bjornsson, and S. Sigmundarson, "Dynamic Control in Real-Time Heuristic Search," *Journal of Artificial Intelligence Research*, vol. 32, no. 1, pp. 419–452, 2008.

[18] A. Blum and M. Furst, "Fast Planning Through Planning Graph Analysis," *Artificial Intelligence*, vol. 90, pp. 1636–1642, 1997.

[19] A. Botea and A. Ciré, "Incremental Heuristic Search for Planning with Temporally Extended Goals and Uncontrollable Events," in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2009, pp. 1647–1652.

[20] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 2, pp. 100—107, 1968.

[21] Y. Chen, Y. Xu, and G. Yao, "Stratified Planning," in *Proceedings of the Internation Joint Conference on Artificifial Intelligence*, 2009, pp. 1665–1670.